Tight Lower Bounds on the Complexity of Derivative Accumulation

Andrew Lyons

Computation Institute, University of Chicago, and
Mathematics and Computer Science Division, Argonne National Laboratory
lyonsam@gmail.com

Theory Seminar
Department of Computer Science, University of Chicago
March 9, 2010

Who Am I?

- B.S. Computer Science, Mathematics (Vanderbilt Univ. 2006)
- Background in graph/order theory, algorithms
- 2007-present: ANL

Who Am I?

- B.S. Computer Science, Mathematics (Vanderbilt Univ. 2006)
- Background in graph/order theory, algorithms
- 2007-present: ANL

Specialized compiler OpenAD (http://www.mcs.anl.gov/OpenAD/) implementing techniques of *automatic* (or *algorithmic*) *differentiation*

Primary application: MITgcm (General Circulation Model) (http://mitgcm.org/)

Motivation: Derivatives are Ubiquitous in Computational Science and Engineering

Examples:

- Derivative-based optimization
- Numerical simulation (sensitivities)

Have code for F,

Want code to compute the value for F and its derivatives F' (at some argument)

A Very High-Level Overview of Computational Derivatives

Divided Differences

- Treat F as a black box
- involves step-size parameter h (inexact, needs tuning)

Symbolic Differentiation (Mathematica, etc.)

- Ignore code for F, treat as a collection of expressions (formulas)
- $\Rightarrow$ produce *formula* for F' from *formula* for F

A Very High-Level Overview of Computational Derivatives

Divided Differences

- Treat F as a black box
- involves step-size parameter h (inexact, needs tuning)

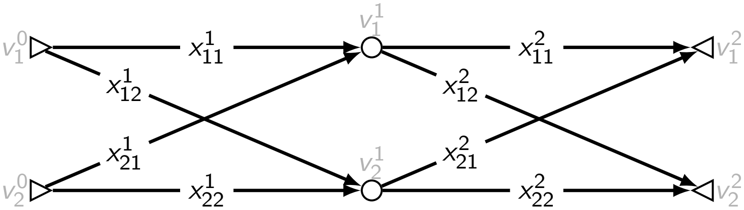
Symbolic Differentiation (Mathematica, etc.)

- Ignore code for F, treat as a collection of expressions (formulas)
- $\Rightarrow$ produce *formula* for F' from *formula* for F

Automatic (Algorithmic) Differentiation

- code for $F \xrightarrow{\text{OpenAD}}$ code for F and $F' \xrightarrow{\text{traditional compiler}}$ machine code
- Considers the code for F as a *circuit*, appends to this a circuit for F'
- Yields *exact derivatives*

The Optimal Structural Derivative Accumulation Problem

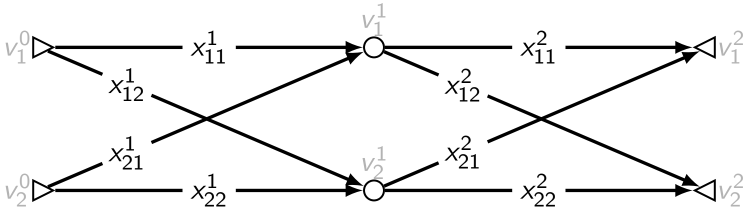


straight-line code $\longrightarrow$ G

Given any DAG G, find optimal way to evaluate

$$\mathcal{J}_{ij}(G) = \sum_{P \in [s_i \rightsquigarrow t_j]} \prod_{(u,v) \in P} x_{uv},$$

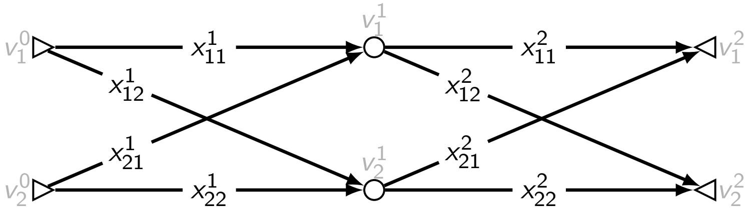
The Optimal Structural Derivative Accumulation Problem



exponential number of terms – easy to evaluate by dynamic programming

Straight-line code (no branches) – is this a toy problem?

The Optimal Structural Derivative Accumulation Problem



$$\mathcal{J}(G) = \begin{bmatrix} x_{11}^1 & x_{12}^1 \\ x_{21}^1 & x_{22}^1 \end{bmatrix} \begin{bmatrix} x_{11}^2 & x_{12}^2 \\ x_{21}^2 & x_{22}^2 \end{bmatrix}$$

What can we hope to say about the complexity of $\mathcal{J}(G)$?
it includes matrix multiplication as a special case

Tight Lower Bounds for Computations over Semirings

We restrict our computation to the real semiring ($\Rightarrow$ monotone circuits)

Theorem (Jerrum/Snir 1982)

$(k-1)n^3$ multiplications are necessary and sufficient to evaluate the product $A^1 A^2 \cdots A^k$ of k dense $n \times n$ matrices over $\langle \mathbb{R}, \times, +, 0, 1 \rangle$.

Tight Lower Bounds for Computations over Semirings

We restrict our computation to the real semiring ($\Rightarrow$ monotone circuits)

Theorem (Jerrum/Snir 1982)

$(k-1)n^3$ multiplications are necessary and sufficient to evaluate the product $A^1 A^2 \cdots A^k$ of k dense $n \times n$ matrices over $\langle \mathbb{R}, \times, +, 0, 1 \rangle$.

For $k = 2$, the above is implied by the following stronger result.

Theorem ((many – Pratt, Paterson, Kerr, Melhorn) 1970's)

If A is an $n_0 \times n_1$ matrix and B is an $n_1 \times n_2$ matrix, then $n_0 n_1 n_2$ multiplications and $n_0(n_1 - 1)n_2$ additions are necessary and sufficient to evaluate AB over any semiring of characteristic zero.

Why Compute Over a Semiring?

Some combination of the following:

- Numerical stability (no run-time checks)
- Seems natural
- Our purview is the *structure* of derivatives and the chain rule
- This structure should certainly have meaning in semirings

Outline

Outline

Computational Model

The real semiring $\langle \mathbb{R}, \times, +, 0, 1 \rangle$

- $\times$ and $+$ are commutative, associative
- $\times$ distributes over $+$
- 1 - multiplicative identity
- 0 - additive identity/multiplicative annihilator
- No additive inverses – no cancellations

Arithmetic Circuits Compute (Collections of) Polynomials

Inputs: indeterminates from X, positive constants from underlying field

Gates: Always indegree 2, of the following two types:

- $\otimes$ gates : Compute the product of their children
- $\oplus$ gates : Compute the sum of their children

Arithmetic Circuits Compute (Collections of) Polynomials

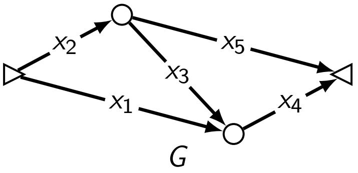
Inputs: indeterminates from X, positive constants from underlying field

Gates: Always indegree 2, of the following two types:

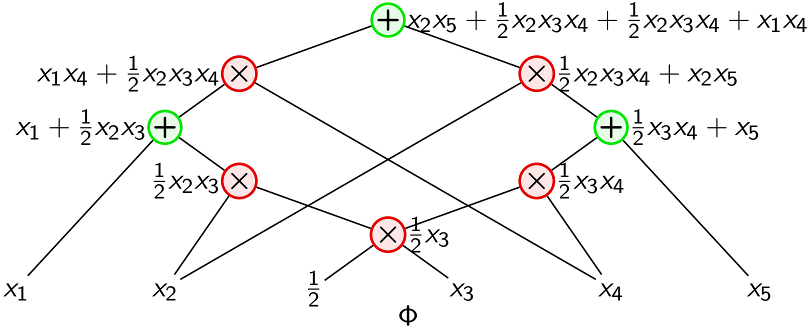
- $\otimes$ gates : Compute the product of their children
- $\oplus$ gates : Compute the sum of their children

Think of polynomials in terms of set of sets representation (monomials and indeterminates)

Arithmetic Circuits Compute (Collections of) Polynomials



$$\mathcal{J}(G) = x_2x_5 + x_2x_3x_4 + x_1x_4$$



Monotone Multilinear Circuits Have Nice Properties

Definition (multilinear polynomial over $\mathbb{R}[X]$)

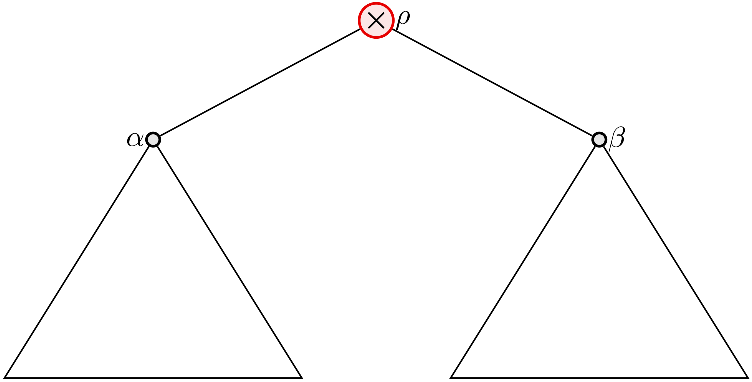
linear in each indeterminate in X

Monotone circuits for multilinear polynomials are multilinear
(Nisan/Wigderson 1995)

Monotone Multilinear Circuits Have Nice Properties

Definition (multiplicatively disjoint circuit)

No indeterminate x has both α and β as an ancestor



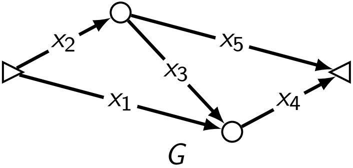
Parse Trees

Definition (Jerrum/Snir1982)

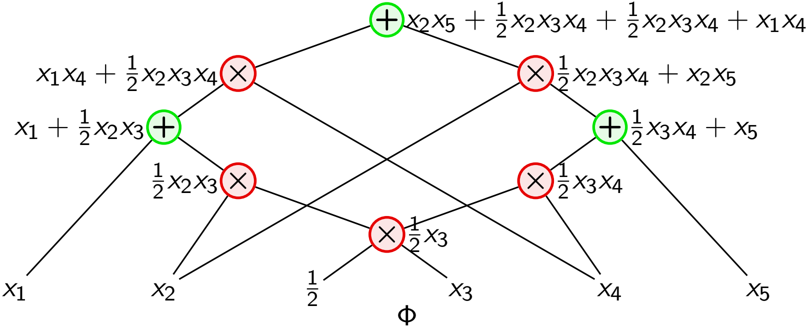
A subcircuit T of Φ is a parse tree of Φ if it satisfies the following conditions:

1. *T contains the (unique) output of Φ.*
2. *If T contains a sum gate σ, then T contains exactly one of the children of σ.*
3. *If T contains a product gate ρ, then T contains both of the children of ρ.*
4. *No proper subtree of T satisfies (i)-(iii).*

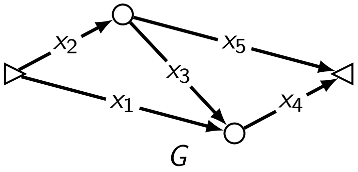
Parse Trees



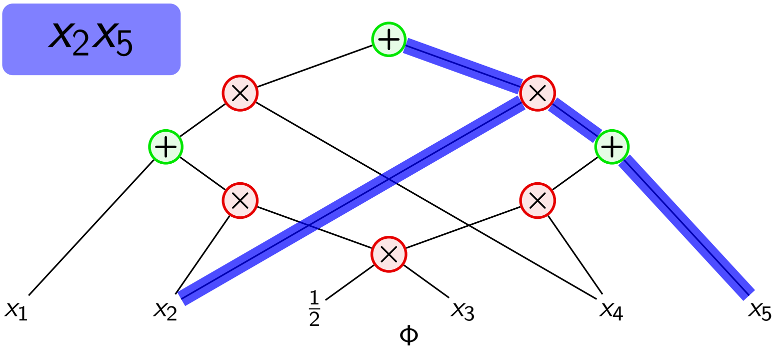
$$\mathcal{J}(G) = x_2x_5 + x_2x_3x_4 + x_1x_4$$



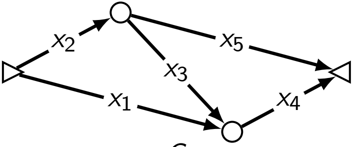
Parse Trees



$$\mathcal{J}(G) = x_2x_5 + x_2x_3x_4 + x_1x_4$$



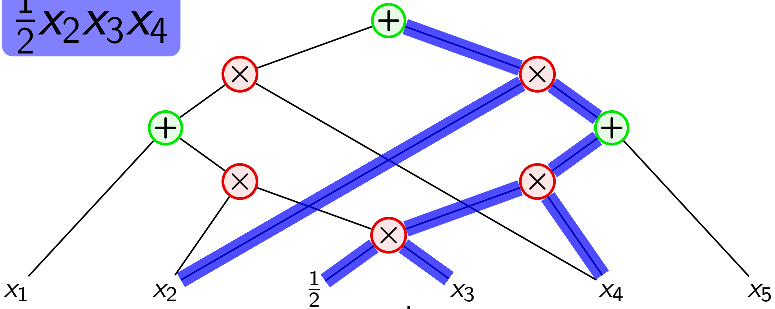
Parse Trees



G

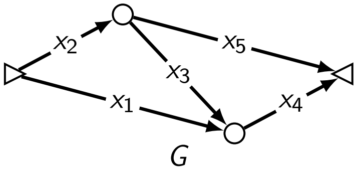
$$\mathcal{J}(G) = x_2x_5 + x_2x_3x_4 + x_1x_4$$

$$\frac{1}{2}x_2x_3x_4$$



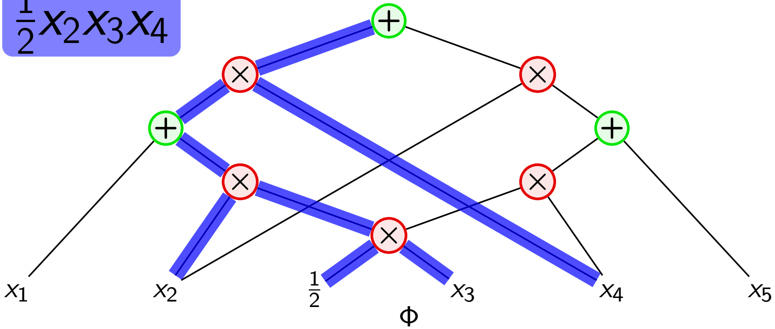
Φ

Parse Trees

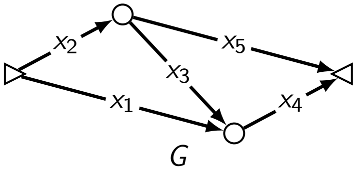


$$\mathcal{J}(G) = x_2x_5 + x_2x_3x_4 + x_1x_4$$

$\frac{1}{2}x_2x_3x_4$

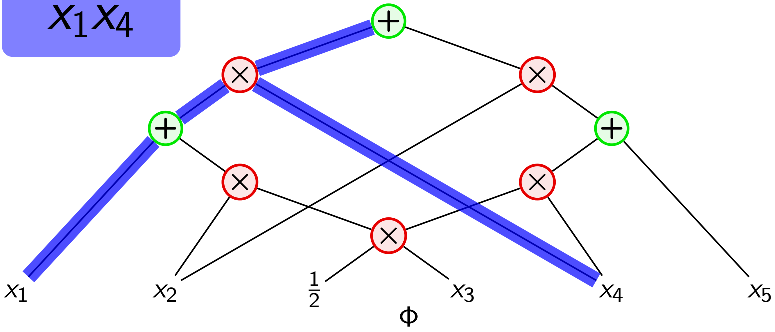


Parse Trees



$$\mathcal{J}(G) = x_2x_5 + x_2x_3x_4 + x_1x_4$$

x_1x_4



Outline

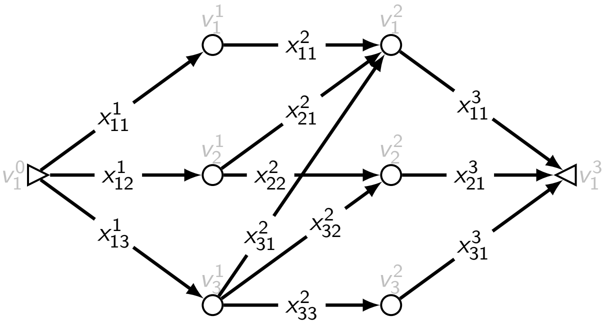
Tight Lower Bounds

Theorem

An optimal arithmetic circuit computing $\mathcal{J}(G)$ can be constructed in polynomial time if G belongs to one of the following classes of DAGs.

- *3-homogeneous st-DAGs*
- *complete st-DAGs*
- *series-parallel st-DAGs*

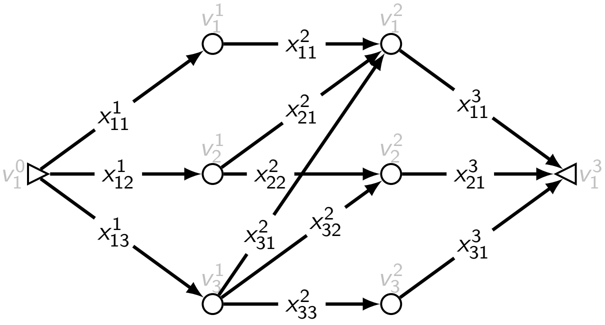
3-homogeneous *st*-DAGs



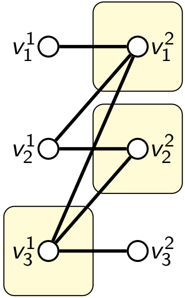
G

$$\mathcal{J}(G) = \underbrace{\begin{bmatrix} x_{11}^1 & x_{12}^1 & x_{13}^1 \end{bmatrix}}_{X^1} \underbrace{\begin{bmatrix} x_{11}^2 & 0 & 0 \\ x_{21}^2 & x_{22}^2 & 0 \\ x_{31}^2 & x_{32}^2 & x_{33}^2 \end{bmatrix}}_{X^2} \underbrace{\begin{bmatrix} x_{11}^3 \\ x_{21}^3 \\ x_{31}^3 \end{bmatrix}}_{X^3}$$

3-homogeneous *st*-DAGs



G



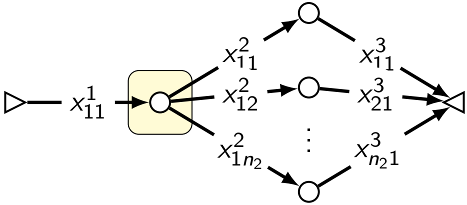
G^2

If G is a 3-homogeneous *st*-DAG, then

$$\mathbf{C}_\times\left(\mathcal{J}(G)\right) = \left|X^2\right| + \tau\left(G^2\right) \quad .$$

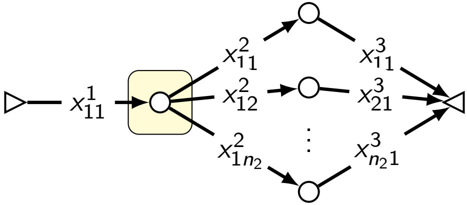
3-homogeneous st-DAGs: The Upper Bound

Let H be a vertex cover of G^2, and assume WLOG that $v_1^1 \in H$

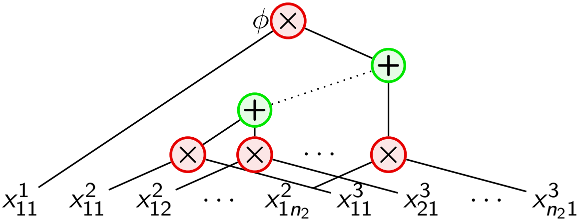


3-homogeneous *st*-DAGs: The Upper Bound

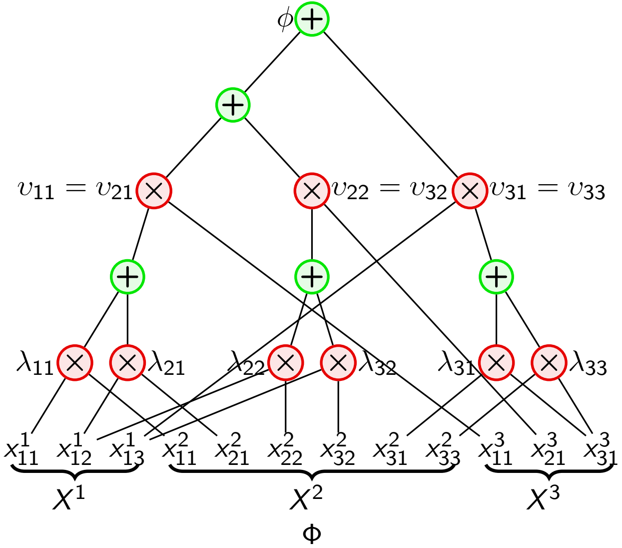
Let H be a vertex cover of G^2, and assume WLOG that $v_1^1 \in H$

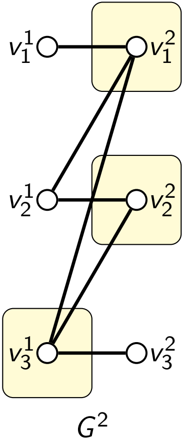


Produce a (sub)circuit for all paths containing x_{11}^1



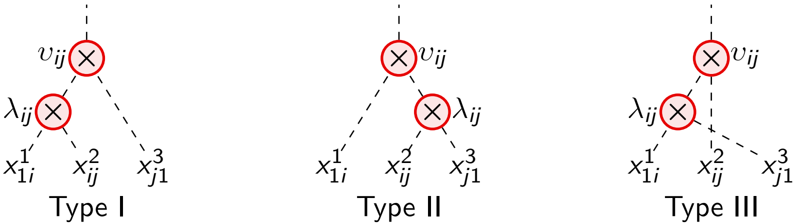
3-homogeneous *st*-DAGs: The Upper Bound





3-homogeneous *st*-DAGs: The Lower Bound

Note 1-1 correspondence between monomials of $\mathcal{J}(G)$ and elements of X^2

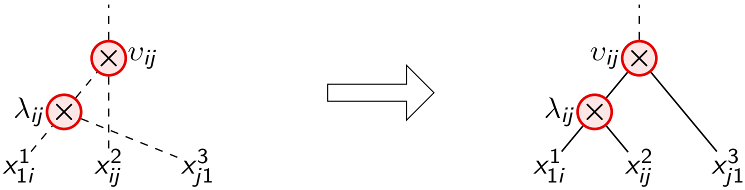


Consider the gates where indeterminates come together

Λ: (the "lower") gates – two indeterminates

Υ: (the "upper") gates – three indeterminates

3-homogeneous *st*-DAGs: The Lower Bound



$$|\Lambda| \geq |X^2|$$

$$|\Upsilon| \geq \tau\left(G^2\right)$$

Lower Bounds via Reduction Rules

We consider local transformations

$$G \longrightarrow G'$$

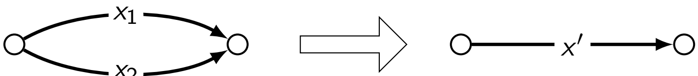
where we can relate the complexity of G to that of G'

In some cases, a sequence

$$G \rightarrow G' \rightarrow \cdots \rightarrow G^{(k-1)} \rightarrow G^{(k)}$$

with $k = O\left(|A(G)|\right)$ reduces the graph to a *single edge*.

Lower Bounds via Reduction Rules: Parallel Arcs



Lemma

$$\mathbf{C}\left(\mathcal{J}(G)\right) = \mathbf{C}\left(\mathcal{J}(G')\right) + 1$$
$$\mathbf{C}_{+}\left(\mathcal{J}(G)\right) = \mathbf{C}_{+}\left(\mathcal{J}(G')\right) + 1$$
$$\mathbf{C}_{\times}\left(\mathcal{J}(G)\right) = \mathbf{C}_{\times}\left(\mathcal{J}(G')\right)$$

Proof.

($\leq$): set $x' = x_1 + x_2$

($\geq$): set $x_1 = 0$ (removes at least one sum gate) $\square$

Lower Bounds via Reduction Rules: Key Lemma

Let (u, v) be an arc in $A(G)$.

Lemma

If there is no alternative path from u to v in G,
then every parent of $x_{uv} \in \Phi$ is a $\otimes$-gate

Proof.

Suppose a sum gate σ has children x_{uv} and β.
For every parse tree that includes x_{uv} there is a corresponding parse tree including β. $\square$

Lower Bounds via Reduction Rules: Arcs in Series

$$\circ \xrightarrow{x_1} \overset{v}{\circ} \xrightarrow{x_2} \circ \quad \Longrightarrow \quad \circ \xrightarrow{\qquad x' \qquad} \circ$$

Lemma

If v has exactly one inedge and exactly one outedge, then

$$\begin{aligned}
\mathbf{C}\left(\mathcal{J}(G)\right) &= \mathbf{C}\left(\mathcal{J}(G')\right) + 1 \\
\mathbf{C}_{+}\left(\mathcal{J}(G)\right) &= \mathbf{C}_{+}\left(\mathcal{J}(G')\right) \\
\mathbf{C}_{\times}\left(\mathcal{J}(G)\right) &= \mathbf{C}_{\times}\left(\mathcal{J}(G')\right) + 1
\end{aligned}$$

Lower Bounds via Reduction Rules: Arcs in Series

$$\circ \xrightarrow{x_1} \overset{v}{\circ} \xrightarrow{x_2} \circ \quad \Longrightarrow \quad \circ \xrightarrow{\quad x' \quad} \circ$$

Lemma

If v has exactly one inedge and exactly one outedge, then

$$\begin{aligned}
\mathbf{C}\left(\mathcal{J}(G)\right) &= \mathbf{C}\left(\mathcal{J}(G')\right) + 1 \\
\mathbf{C}_{+}\left(\mathcal{J}(G)\right) &= \mathbf{C}_{+}\left(\mathcal{J}(G')\right) \\
\mathbf{C}_{\times}\left(\mathcal{J}(G)\right) &= \mathbf{C}_{\times}\left(\mathcal{J}(G')\right) + 1
\end{aligned}$$

Proof.

$\leq$: set $x' = x_1 \times x_2$

$\geq$: set $x_1 = 1$ (remove at least one $\otimes$-gate)

$\square$

Lower Bounds via Reduction Rules: Series-Parallel *st*-DAGs

Definition

A single isolated edge is a series-parallel st-DAG.
If G_1, G_2 are series-parallel st-DAGs, then so is their…

series composition: *identify the sink of G_1 with the source of G_2*

parallel composition: *identify the two sources, identify the two sinks*

Lower Bounds via Reduction Rules: Series-Parallel *st*-DAGs

Definition

A single isolated edge is a series-parallel st-DAG.
If G_1, G_2 are series-parallel st-DAGs, then so is their...

series composition: *identify the sink of G_1 with the source of G_2*

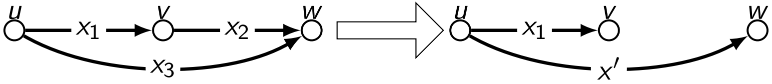
parallel composition: *identify the two sources, identify the two sinks*

Theorem

The following are equivalent.

- *G is a series-parallel st-DAG*
- *G can be reduced to a single edge by a sequence of series and parallel reduction rule applications*
- *there is a circuit for $\mathcal{J}(G)$ that is tree structured (like a formula)*

Lower Bounds via Reduction Rules: Complete *st*-DAGs



Lemma

If v has exactly one inedge and there is no alternative path from v to w, then

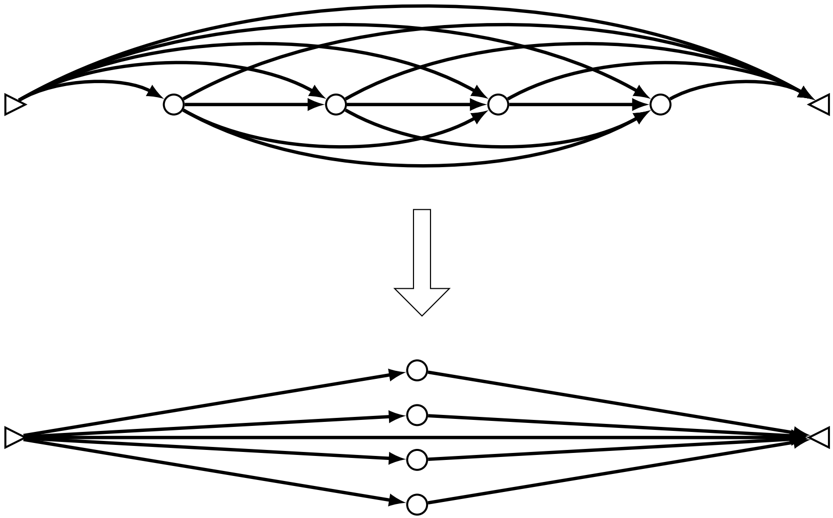
$$\mathbf{C}\left(\mathcal{J}(G)\right) = \mathbf{C}\left(\mathcal{J}(G')\right) + 2$$
$$\mathbf{C}_{+}\left(\mathcal{J}(G)\right) = \mathbf{C}_{+}\left(\mathcal{J}(G')\right) + 1$$
$$\mathbf{C}_{\times}\left(\mathcal{J}(G)\right) = \mathbf{C}_{\times}\left(\mathcal{J}(G')\right) + 1$$

Proof.

$(\leq)$: set $x' = x_3 + (x_1 \times x_2)$

$(\geq)$: set $x_2 = 0$ (removes at least one $\otimes$-gate and at least one $\oplus$-gate) $\square$

Lower Bounds via Reduction Rules: Complete *st*-DAGs



Lower Bounds via Reduction Rules: Comments

Optimality-preserving reduction rules should be applied whenever possible

We can turn any DAG into a homogeneous DAG by *subdividing arcs* (series reduction rule)

All of our reduction rules run in polynomial time.

future work: could these rules (or similar) imply a polynomial-size kernel?

Outline

Discussion of Results

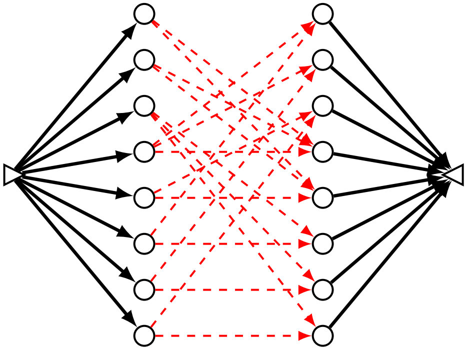
What have we seen so far?

- homogeneous DAGs correspond to iterated sparse matrix multiplication
- finding an optimal circuit for a 3-homogeneous *st*-DAG $\Leftrightarrow$ bipartite vertex cover
- Lower bounds via *reduction rules* for series-parallel and complete *st*-DAGs

Progress towards to original problem (Optimal Structural Derivative Accumulation)?

Complexity of Circuit Minimization

The problem becomes NP-hard when some subset of the edges may be labeled with the multiplicative unit “1”.



G

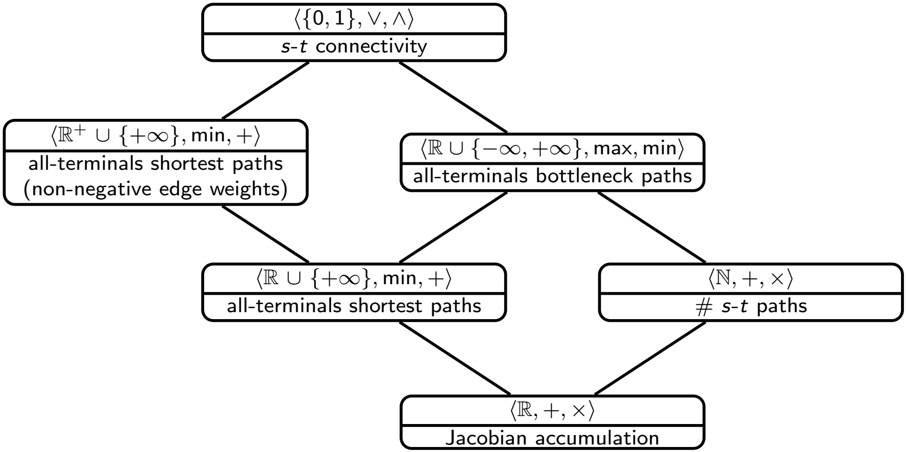
$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

X^2

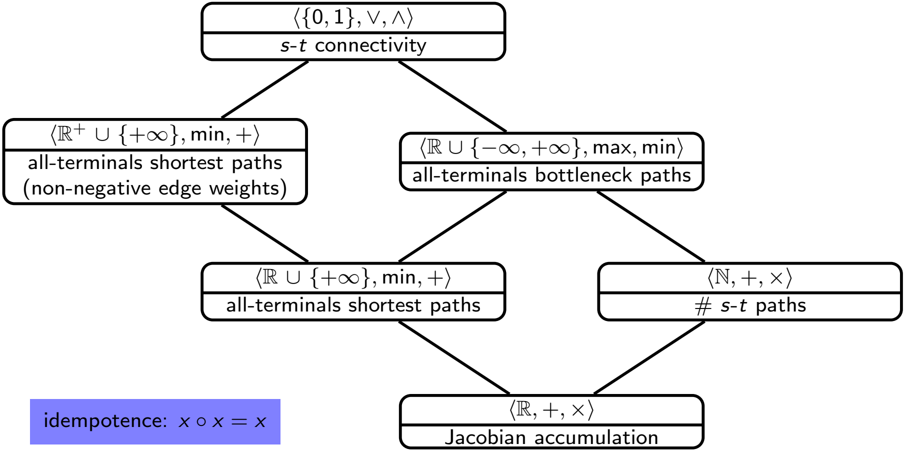
$\Rightarrow$ bilinear forms with $\{0, 1\}$ constants

NP-hard via biclique cover (Gonzalez and JáJá, 1980)

Computing Polynomial Functions over Different Semirings



Computing Polynomial Functions over Different Semirings



idempotence: $x \circ x = x$

The Power of Constants

constant terms

$$(1 + x_a)(x_b + x_c) = x_b + x_c + x_a x_b + x_a x_c$$

this does not apply for *homogeneous* polynomials, and it also doesn't apply for "path polynomials"

Lemma

The parent of every constant input is a product gate.

Proof.

(Same as for edges with no alternative path.) □

The Power of Constants: Monotone Multilinear Circuits Without Constants are Even Nicer

scaling indeterminates by constants

$$x_1 + ax_2 + (1-a)x_2 + x_3$$

why is it useful to have constant-free circuits?

The Power of Constants

$$\mathcal{R} = \langle \mathbb{R}, +, \times, 0, 1 \rangle \quad \mathcal{M} = \langle \mathbb{R} \cup \{+\infty\}, \min, +, +\infty, 0 \rangle$$

Theorem (Jerrum/Snir 1982)

If p is a multilinear polynomial, then

$$\mathbf{C}^{\mathcal{M}}(p) = \mathbf{C}^{\mathcal{R}}(p)$$
$$\mathbf{C}_{\times}^{\mathcal{M}}(p) = \mathbf{C}_{\times}^{\mathcal{R}}(p)$$
$$\mathbf{C}_{+}^{\mathcal{M}}(p) = \mathbf{C}_{+}^{\mathcal{R}}(p)$$

Optimal Circuits are Constant-Free

Conjecture

Let p be monic, multilinear.
If p is homogenous or p is the path polynomial of some st-DAG, then every optimal arithmetic circuit computing p over $\langle \mathbb{R}, +, \times \rangle$ is constant-free.

Proof.

If a monotone idempotent circuit computes a monic multilinear polynomial, then we can remove the constants $\square$

The Power of Constants

$$\mathcal{R} = \langle \mathbb{R}, +, \times, 0, 1 \rangle, \quad \mathcal{M}^+ = \langle \mathbb{R}^+ \cup \{+\infty\}, \min, +, +\infty, 0 \rangle$$

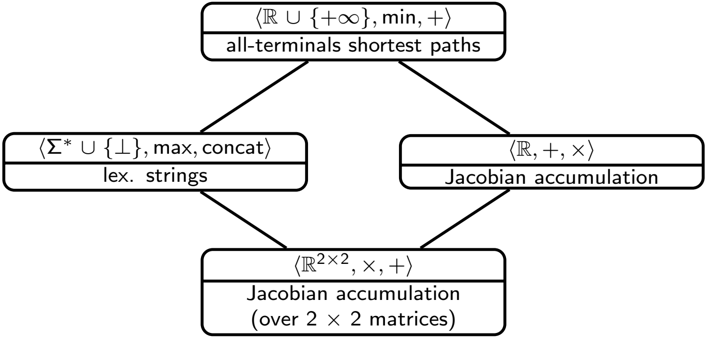
Theorem (Jerrum/Snir 1982)

If p is a homogeneous multilinear polynomial, then

$$\mathbf{C}^{\mathcal{M}^+}(p) = \mathbf{C}^{\mathcal{R}}(p)$$
$$\mathbf{C}_{\times}^{\mathcal{M}^+}(p) = \mathbf{C}_{\times}^{\mathcal{R}}(p)$$
$$\mathbf{C}_{+}^{\mathcal{M}^+}(p) = \mathbf{C}_{+}^{\mathcal{R}}(p)$$

Note here we have *absorption*: $\min(a, a + b) = a$

The Power of Commutativity



Conjecture (Griewank/Naumann)

Commutativity has no power for evaluating $\mathcal{J}(G)$

All our upper bounds use noncommutative circuits

Acknowledgements

- Jean Utke/Paul Hovland/Ilya Safro (ANL)
- Uwe Naumann (RWTH Aachen)
- Andreas Griewank (Humboldt Berlin)
- Sasha Razborov/Raghav Kulkarni (Chicago)
- Andrew Cone (Chicago alum)

Thanks!

Questions?